AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A system that facilitates a performance enhancement in message-based computing, comprising one or more computing devices configured with:

a performance-based event transmission interface across which a request from a source is transmitted to a destination; and

a filter component in communication with the destination that dynamically determines which response information is relevant in order to filter out any excess response information that is deemed irrelevant, such that ~~allows~~ only ~~one or more~~ relevant responses from the destination ~~to transition~~ are transmitted ~~the interface~~ to the source via the event transmission interface, wherein the filter component is part of at least one of native code and managed code and wherein irrelevant responses are discarded;

wherein the source notifies a native layer that an event handler has attached or detached.

2. (Original) The system of claim 1, the source comprises at least one of native code and managed code, and the destination comprises at least one of native code and managed code.

3. (Canceled)

4. (Previously Presented) The system of claim 1, the destination issues a callback to the source, in response the filter component permits only a relevant response to the callback to be transmitted to the destination.

5. (Original) The system of claim 1, the filter component is integrated into an operating system.

6. (Original) The system of claim 1 is employed in a small footprint execution environment that has reduced resources.

7. (Original) The system of claim 1, the source is part of a managed code framework that includes a graphical user interface (GUI) application that transmits an event for processing by the destination, which destination is part of native code.

8. (Original) The system of claim 1, further comprising a classifier that makes an inference about processes that can be automatically performed.

9. (Currently Amended) A computer readable storage medium having stored thereon computer executable instructions for carrying out the system of claim 1.

10. (Original) A computer that employs the system of claim 1.

11. (Original) A server that employs the system of claim 1.

12. (Original) The system of claim 1, the source is associated with a GUI such that the request receives only the one or more responses from a graphics/windowing/events system that are relevant to the request.

13. (Canceled)

14. (Original) The system of claim 1, the filter component is notified by the source when an event handler has attached to or detached from a source object, and forwards the one or more responses only when the associated event handlers are attached.

15. (Original) The system of claim 1, the source can dynamically inspect at least one of properties, methods, and events implemented on a source object.

16. (Original) The system of claim 1, the source utilizes reflection during initialization of an object to determine the presence of a custom object.

17. (Original) The system of claim 1, the source utilizes reflection during initialization of an object to determine if message handling has been modified in a custom object.

18. (Original) The system of claim 1, the source utilizes type introspection to determine the presence of a custom object, in response to which the destination is notified that a message associated with the custom object is of interest and will be forwarded from the destination for a lifetime of the custom object.

19. (Currently Amended) A system that facilitates a performance enhancement in message-based computing, comprising one or more computing devices configured with:

a managed code framework that generates a request;

a native code framework that receives the request and issues one or more responses thereto;

a performance-based event transmission interface between the managed code framework and the native code framework across which the request is passed and the one or more responses are transmitted; and

a filter component in communication with the native code framework that dynamically determines which response information is relevant in order to filter out any excess response information that is deemed irrelevant, such that ~~allows~~ only ~~one or more~~ relevant responses of the native code framework ~~to transition~~ are transmitted ~~the interface~~ to the managed code framework via the event transmission interface;

wherein the filter component is notified by the managed code framework when an event handler has at least one of registered and unregistered from a managed object and forwards the one or more responses only when the associated event handlers are registered.

20. (Original) The system of claim 19, the filter component is part of the native code framework.

21. (Original) The system of claim 19, the filter component only allows the one or more responses across the interface that are relevant to the request.

22. (Original) The system of claim 19, further comprising a managed code filter component that is part of the managed code framework.

23. (Original) The system of claim 19, further comprising a managed code filter component that is part of the managed code framework, which managed code filter processes a callback from the native code framework and only forwards responses from the managed code to the native code that are relevant to the callback.

24. (Original) The system of claim 19, the filter component only processes events that are registered.

25. (Canceled)

26. (Original) The system of claim 19, the managed code framework source utilizes type introspection during initialization of an object to determine the presence of a custom object.

27. (Original) The system of claim 19, the managed code framework utilizes type introspection during initialization of an object to determine if message handling has been modified in a custom object.

28. (Original) The system of claim 19, the managed code framework utilizes type introspection to determine the presence of a custom object, in response to which the native code framework is notified that a message associated with the custom object is of interest and will be forwarded from the native code framework for a lifetime of the custom object.

29. (Currently Amended) A computer-readable storage medium tangibly embodying thereon computer-executable instructions for performing a method of managing messages across a performance-based interface, the method comprising:

receiving the performance-based event transmission interface across which a request from a source is transmitted to a destination;

returning one or more responses to the request from the destination;

determining which response information is relevant in order to filter out any excess response information that is deemed irrelevant;

filtering the one or more responses to allow only responses that are relevant to the request to be transmitted from the destination to the source via the event transmission interface; and

transmitting the relevant responses across the interface to the source[[~]];

wherein the source notifies a native layer that an event handler has attached or detached.

30. (Currently Amended) The computer-readable storage medium of claim 29, the act of filtering occurs on the destination side of the interface.

31. (Currently Amended) The computer-readable storage medium of claim 29, the method further comprising at least one of the acts of:

transmitting the relevant responses according to priority criteria;

transmitting a request from the destination to the source;

filtering the one or more responses to allow only responses from the source that are relevant to the request; and

transmitting the relevant responses across the interface to the destination.

32. (Currently Amended) A method of managing messages across a performance-based interface, comprising

using one or more processors to perform the following computer-executable acts:

receiving the performance-based event transmission interface between a managed code and a native code;

registering an event handler for an event of a managed object;

notifying the native code that the event handler is registered;

returning a one or more responses of the native code associated with the event;

determining which response information is relevant in order to filter out any excess response information that is deemed irrelevant;

filtering the one or more responses of the native code to determine the relevant responses associated with event, such that only responses that are relevant to the request are allowed to be transmitted from the destination to the source via the event transmission interface; and

transmitting the relevant responses across the interface to the managed code only when the associated event handler is registered.

33. (Previously Presented) The method of claim 32, further comprising tracking the attached event with the native code.

34-37. (Canceled)

38. (Currently Amended) A system that facilitates the management of messages across a performance-based interface, comprising:

a processor;

system memory;

means for receiving the performance-based event transmission interface between a managed code and a native code;

means for registering an event handler for an event of a managed object;

means for notifying the native code that the event handler is registered;

means for returning a one or more responses of the native code associated with the event;

means for determining which response information is relevant in order to filter out any excess response information that is deemed irrelevant;

means for filtering the one or more responses of the native code to determine the relevant responses associated with event, such that only responses that are relevant to the request are allowed to be transmitted from the destination to the source via the event transmission interface; and

means for transmitting the relevant responses across the interface to the managed code only when the associated event handler is registered.

39. (Original) The system of claim 38, further comprising means for unregistering the one or more event handlers when the associated events have expired.

40. (Previously Presented) The method of claim 32, further comprising:

processing a request, issued by an application, associated with the event in accordance with a destination process;

unregistering the event;

notifying an event filter that the event is unregistered; and

signaling the application that the event has been unregistered.